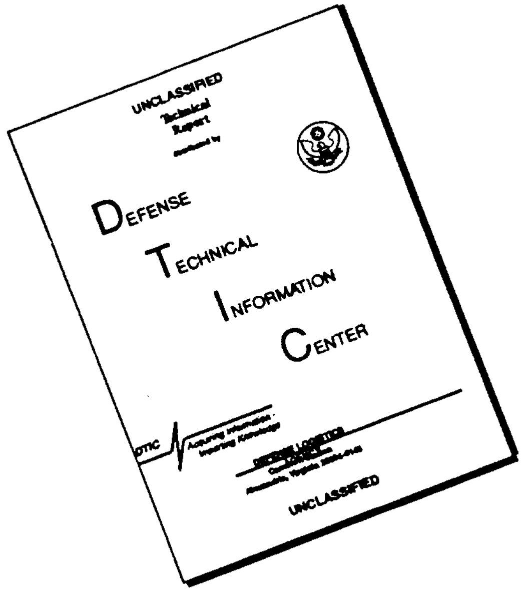
Evaluation of Sampling for Data Mining of Association Rules

M.J. Zaki, S. Parthasarathy, W. Li, and M. Ogihara

Technical Report 617
May 1996

UNIVERSITY OF
ROCHESTER
COMPUTER SCIENCE

DISCLAIMER NOTICE



THIS DOCUMENT IS BEST QUALITY AVAILABLE. THE COPY FURNISHED TO DTIC CONTAINED A SIGNIFICANT NUMBER OF PAGES WHICH DO NOT REPRODUCE LEGIBLY.

Evaluation of Sampling for Data Mining of Association Rules *

Mohammed Javeed Zaki, Srinivasan Parthasarathy
Wei Li, Mitsunori Ogihara
{zaki,srini,wei,ogihara}@cs.rochester.edu

The University of Rochester
Computer Science Department
Rochester, New York 14627

Technical Report 617

May 1996

Abstract

Data mining is an emerging research area, whose goal is to extract significant patterns or interesting rules from large databases. High-level inference from large volumes of routine business data can provide valuable information to businesses, such as customer buying patterns, shelving criterion in supermarkets and stock trends. However, many algorithms proposed for data mining of association rules make repeated passes over the database to determine the commonly occurring *itemsets* (or set of items). For large databases, the I/O overhead in scanning the database can be extremely high.

In this paper we show that random sampling of transactions in the database is an effective method for finding association rules. Sampling can speed up the mining process by more than an order of magnitude by reducing I/O costs and drastically shrinking the number of transaction to be considered. We may also be able to make the sampled database resident in main-memory. Furthermore, we show that sampling can accurately represent the data patterns in the database with high confidence. We experimentally evaluate the effectiveness of sampling on three databases.

Keywords: Data Mining, Association Rules, Random Sampling, Chernoff bounds.

The University of Rochester Computer Science Department supported this work.

*This work was supported in part by an NSF Research Initiation Award (CCR-9409120) and ARPA contract F19628-94-C-0057.

REPORT DOCUMENTATION PAGE

Form Approved

OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)

2. REPORT DATE
May 1996

3. REPORT TYPE AND DATES COVERED
technical report

4. TITLE AND SUBTITLE
Evaluation of Sampling for Data Mining of Association Rules

5. FUNDING NUMBERS
ARPA F19628-94-C-0057

6. AUTHOR(S)
M.J. Zaki, S. Parthasarathy, W. Li, and M. Ogihara

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESSES
Computer Science Dept.
734 Computer Studies Bldg.
University of Rochester
Rochester NY 14627-0226

8. PERFORMING ORGANIZATION

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESSES(ES)
ARPA
3701 N. Fairfax Drive
Arlington VA 22203

10. SPONSORING / MONITORING AGENCY REPORT NUMBER
TR 617

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Distribution of this document is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)
(see title page)

14. SUBJECT TERMS
data mining; association rules; random sampling; Chernoff bounds

15. NUMBER OF PAGES
13 pages

16. PRICE CODE
free to sponsors; else $2.00

17. SECURITY CLASSIFICATION OF REPORT
unclassified

18. SECURITY CLASSIFICATION OF THIS PAGE
unclassified

19. SECURITY CLASSIFICATION OF ABSTRACT
unclassified

20. LIMITATION OF ABSTRACT
UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

1 Introduction

With large volumes of routine business data having been collected, business organizations are increasingly turning to the extraction of useful information from such databases. Such high-level inference process may provide information on customer buying patterns, shelving criterion in supermarkets, stock trends, etc. Data mining is an emerging research area, whose goal is to extract significant patterns or interesting rules from such large databases.

Data mining is in fact a broad area which combines research in machine learning, statistics and databases. It can be broadly classified into three main categories: Classification – finding rules that partition the database into disjoint classes; Sequences – extracting commonly occurring sequences in temporal data; and Associations – find the set of most commonly occurring groupings of items. In this paper we will concentrate on data mining for association rules.

The problem of mining association rules over *basket* data was introduced in [1]. Basket data usually consists of a record per customer with a transaction date, along with items bought by the customer. An example of an association rule over such a database could be that 80% of the customers that bought bread and milk, also bought eggs. The data mining task for association rules can be broken into two steps. The first step consists of finding all the sets of items, called as *itemsets*, that occur in the database with a certain user-specified frequency, called *minimum support.* Such itemsets are called *large* itemsets. An itemset of k items is called a *k-itemset.* The second step consists of forming implication rules among the large itemsets found in the first step.

Many algorithms have been proposed for discovering sets of items with minimum support or the large itemsets. The general structure of these algorithms is that during the initial pass over the database the support for all single items (1-itemsets) is counted. The large 1-itemsets are used to generate candidate 2-itemsets. The database is scanned again to obtain occurrence counts for the candidates, and the large 2-itemsets are selected for the next pass. This iterative process is repeated for $k = 3, 4, \cdots$, till there are no more large k-itemsets to be found. For disk resident databases, the I/O overhead in scanning the database for the count of itemsets during each iteration (or a set of iterations) can be extremely high for large databases.

Random sampling is a technique of selecting a small number of units out of the total. Sampling can be utilized when approximate rules would suffice, or for interactive data mining, where the user can gather quick preliminary information about the database. This may help the user to direct the data mining by refining the criterion for "interesting" rules. In this paper we show that random sampling of transactions in the database is an effective way for finding association rules. Sampling can reduce I/O costs by drastically shrinking the number of transaction to be considered, and we may be able to make the database resident in main memory. We show that sampling can speed up the mining process by more than an order of magnitude, and it can provide great accuracy with respect to the association rules.

The rest of the paper is organized as follows. In the next section we formally present the problem of finding all sets of items with minimum support. In section 3 we present an analysis of random sampling from databases. The effectiveness of sampling is experimentally analyzed in section 4. Section 5 presents our conclusions.

2 Data Mining for Association Rules

We now present the formal statement of the problem of mining association rules over basket data. The discussion below closely follows that in [1, 2].

Let $\mathcal{I} = \{i_1, i_2, \cdots, i_m\}$ be a set of m distinct attributes, also called *items*. Each transaction T in the database $\mathcal{D}$ of transactions, has a unique identifier TID, and *contains* a set of items, such that $T \subseteq \mathcal{I}$. An *association rule* is an expression $A \Rightarrow B$, where itemsets $A, B \subset \mathcal{I}$, and $A \cap B = \emptyset$. Each itemset is said to have a *support* s if $s\%$ of the transactions in $\mathcal{D}$ contain the itemset. The association rule is said to have *confidence* c if $c\%$ of the transactions that contain A also contain B, i.e., $c = support(A \cup B)/support(A)$, i.e., the conditional probability that transactions contain the itemset B, given that they contain itemset A. For example, we may have that 80% of the customers that bought bread and milk also bought eggs. The number 80% is the confidence of the rule, the support of the rule is $support(A \cup B)$. Data mining of association rules from such databases consists of finding the set of all such rules which meet the user-specified minimum confidence and support values.

The task of data mining for association rules can be broken into two steps:

1. Find all the large k-itemsets for $k = 1, 2, \cdots$.

2. Generate rules from these large itemsets. Given that X is a large k-itemset, for every non-empty subset $A \subset X$, a rule of the form $A \Rightarrow B$ is generated, where $B = X - A$, and provided that this rule has the required confidence. We refer the reader to [2] for more detail on rule generation. Henceforth, we will deal only with the first step.

Many algorithms for finding large itemsets have been proposed in the literature since the introduction of this problem in [1] (AIS algorithm). In [7] a pass minimization approach was presented, which uses the idea that if an itemset belongs to the set of large $(k+e)$-itemsets, then it must contain $\binom{k+e}{k}$ k-itemsets. The *Apriori* algorithm [2] also uses the property that any subset of a large itemset must itself be large. These algorithms had performance superior to AIS. Newer algorithms with better performance than *Apriori* were presented in [9, 10]. The DHP algorithm [9] uses a hash table in pass k to do efficient pruning of $(k + 1)$-itemsets. The *Partition* algorithm [10] minimizes I/O by scanning the database only twice. In the first pass it generates the set of all potentially large itemsets, and in the second pass the support for all these is measured. The above algorithms are all specialized black-box techniques which do not use any database operations. Algorithms using only general-purpose DBMS systems and relational algebra operations have also been proposed [5, 6].

In this paper we will use the *Apriori* algorithm to evaluate the effectiveness of sampling for data mining. We would like to point out that our results are about sampling, and as such independent of, and equally applicable to any algorithm for mining of association rules. We now present a brief discussion of the *Apriori* algorithm.

2.1 The *Apriori* Algorithm

The naive method of finding large itemsets would be to generate all the 2^m subsets of the universe of m items, count their support by scanning the database, and output those meeting minimum support criterion. It is not hard to see that the naive method exhibits complexity exponential in m, and is quite impractical. *Apriori* follows the basic iterative structure discussed earlier. However the key observation used is that any subset of a large itemset must also be large. During each iteration of the algorithm only candidates found to be large in the previous iteration are used to generate a new candidate set to be counted during the current iteration. A pruning step eliminates any candidate which has a small subset. *Apriori* also uses specialized data structures to speed up the counting and pruning (Hash trees and Hash Tables, respectively.) The algorithm terminates at step t, if there are no large t-itemsets. Let L_k denote the set of Large k-itemsets and C_k the set of candidate k-itemsets. The general structure of the algorithm is given in figure 1. We refer the reader to [2] for more detail on *Apriori*, and its performance characteristics.

```
L_1 = {large 1-itemsets };
for (k = 2; L_{k-1} ≠ ∅; k++)
    C_k = Set of New Candidates;
    for all transactions t ∈ D
        for all k-subsets s of t
            if (s ∈ C_k) s.count++;
    L_k = {c ∈ C_k | c.count ≥ minimum support};
Set of all large itemsets = ∪_k L_k;
```

Figure 1: The *Apriori* Algorithm

We now present a simple example of how *Apriori* works. Let the database, $\mathcal{D}$ =

$\{T_1 = (1,4,5), T_2 = (1,2), T_3 = (3,4,5), T_4 = (1,2,4,5)\}$. Let the minimum support value $MS = 2$. Running through the iterations, we get

$$\begin{aligned}
L_1 &= \{\{1\},\{2\},\{4\},\{5\}\} \\
C_2 &= \{\{1,2\},\{1,4\},\{1,5\},\{2,4\},\{2,5\},\{4,5\}\} \\
L_2 &= \{\{1,2\},\{1,4\},\{1,5\},\{4,5\}\} \\
C_3 &= \{\{1,4,5\}\} \\
L_3 &= \{\{1,4,5\}\}
\end{aligned}$$

Note that while forming C_3 by joining L_2 with itself, we get three potential candidates, {1,2,4},{1,2,5}, and {1,4,5}. However only {1,4,5} is a true candidate, and the first two are eliminated in the pruning step, since they have a 2-subset which is not large (the 2-subset {2,4}, and {2,5} respectively).

3 Random Sampling for Data Mining

Random sampling is a method of selecting n units out of a total N, such that every one of the C_n^N distinct samples has an equal chance of being selected. If a drawn unit is removed from further consideration, it leads to random sampling *without replacement*. On the other hand, if at any drawing, each unit has an equal chance of being selected, no matter how many times it has been drawn, it leads to random sampling *with replacement*. An efficient algorithm for sequential random sampling without replacement is presented in [11].

Random sampling from databases has been successfully used in query size estimation. This may assist the user by providing approximate answers. Such information can be used for statistical analyses of databases, where approximate answers would suffice. It may also be used to estimate selectivities or intermediate result sizes for query optimization [8]. The application of sampling for mining association rules was suggested in [7]. In this paper we extend their suggestion by evaluating the effectiveness of sampling in practice.

3.1 Binomial Distribution

A *Bernoulli trial* is an experiment with only two outcomes. Namely, *success*, which occurs with probability p, and *failure* which occurs with probability $q = 1 - p$. Let X denote a random variable giving the number of successes in n independent Bernoulli trials. The probability distribution of X is given by the *Binomial distribution*,

$$b(k; n, p) = P(X = k) = C_k^n p^k q^{n-k}, \text{for } x = 0, 1, \cdots, n$$

The probability that X is larger than m is given by $P(X \geq m) = \sum_{j=m}^{n} C_j^n p^j q^{n-j}$.

3.2 Chernoff Bounds

Let τ denote the support of an itemset I. We want to select n transactions out of the total N in the Database $\mathcal{D}$. Let the random variable $X_i = 1$ if the i-th transaction contains the itemset I ($X_i = 0$, otherwise). Clearly, $P(X_i = 1) = \tau$ for $i = 1, 2, \cdots n$. We further assume that all $X_1, X_2, \cdots, X_n$ are independent 0-1 random variables. The random variable $\mathbf{X}$ giving the number of transactions in the sample containing the itemset I, has a binomial distribution of n trials, with the probability of success τ (note: the correct distribution for finite populations is the *Hypergeometric distribution*, although the Binomial distribution is a satisfactory approximation [3]). Moreover, $\mathbf{X} = \sum_i^n X_i$, and the expected value of $\mathbf{X}$ is given as $\mu = E[\mathbf{X}] = E[\sum_{i=1}^n X_i] = \sum_{i=1}^n E[X_i] = n\tau$, since $E[X_i] = 0 \cdot P(X = 0) + 1 \cdot P(X = 1) = \tau$.

For any positive constant, $0 \leq \epsilon \leq 1$, the Chernoff bounds [4] state that

$$P(\mathbf{X} \leq (1-\epsilon)n\tau) \leq e^{-\epsilon^2 n\tau/2} \quad (1)$$

$$P(\mathbf{X} \geq (1+\epsilon)n\tau) \leq e^{-\epsilon^2 n\tau/3} \quad (2)$$

Chernoff bounds provide information on how close is the actual occurrence of an itemset in the sample, as compared to the expected count in the sample. This aspect, which we call as the *accuracy* of a sample, is given by $1-\epsilon$. The bounds also tell us the probability that a sample of size n will have a given accuracy. We call this aspect the *confidence* of the sample (defined as 1 minus the expression on the right hand size of the equations). Chernoff bounds give us two set of confidence values. Equation 1 gives us the lower bound – the probability that the itemset occurs less often than expected (by the amount $n\tau\epsilon$), while equation 2 gives us the upper bound – the probability that the itemset occurs more often than expected, for a desired accuracy. A low probability corresponds to high confidence, and a low ϵ corresponds to high accuracy. It is not hard to see that there is a trade-off between accuracy and confidence for a given sample size. This can been seen immediately, since $\epsilon = 0$ maximizes the right hand side of equations 1,2, while $\epsilon = 1$ minimizes it.

Sample Size Selection

Given that we are willing to accommodate a certain accuracy, $\mathcal{A} = 1 - \epsilon$, and confidence $\mathcal{C} = 1 - c$ of the sample, the Chernoff bounds can be used to obtain a sample size. We'll show this for equation 1, by plugging in $c = e^{-\epsilon^2 n\tau/2}$, to obtain

$$n = -2\ln(c)/(\tau\epsilon^2) \quad (3)$$

If we know the support for each itemset we could come up with a sample size n_I for each itemset I. We would still have the problem of selecting a single sample size from among the n_I. One simple heuristic is to use the user specified minimum support threshold for τ. The rationale is that by using this we guarantee that the sample size contains all the large itemsets contained in the original database. For example, let the total transactions in the original database $N = 3,000,000$. Let's say we desire a confidence $\mathcal{C} = 0.9(c = 0.1)$, and an accuracy $\mathcal{A} = 0.99(\epsilon = 0.01)$. Let the user specified support threshold be 1%. Using these values in equation 3, we obtain a sample size of $n = 4,605,170$. This is even greater than the original database! The problem is that the sample size expression is independent of the original database size. Moreover the user specified threshold is also independent of the actual itemset support in the original database. Hence, using this value may be too conservative, as shown above. In the next section we will compare experimental results obtained versus the theoretical predictions using Chernoff bounds.

4 Experimental Evaluation

In this section we describe the experiments conducted in order to determine the effectiveness of sampling. We demonstrate that it is a reasonably accurate technique in terms of the associations generated by the sample, as compared to the associations generated by the original database. At the same time sampling can help reduce the execution time by more than an order of magnitude.

4.1 Experimental Framework

The base algorithm for generating associations is *Apriori*. All experiments were conducted on a DEC alpha processor. We used three different databases to evaluate the effectiveness of sampling. These are:

- **SYNTH:** This is a database of synthetic transactions, based on the one used in [2]. This database mimics the transactions in a retailing environment. Each transaction has a unique ID followed by a list of items bought in that transaction. The data-mining provides information about the set of items generally bought together. We obtained the database $T10.I4.D250K$, by setting the number of transactions $|\mathcal{D}| = 250000$, average transaction size $|T| = 10$, average maximal potentially large itemset size $|I| = 4$, number of maximal potentially large itemsets $|L| = 2000$, and the number of items $N = 1000$. We refer the reader to [2] for more detail on the database generation.

- **ENROLL:** This is a database of student enrollments for a particular graduating class. Each transaction consists of a student ID followed by information on the college, major, department, semester, and a list of courses taken during that semester. The data-mining answers questions concerning the set of large courses, and their relationship with the other attributes. There are 39624 transactions, 3581 items and the average transaction size is 9.

- **TRBIB:** This is a database of the locally available technical report bibliographies in computer science. Each item is a key-word which appears in a paper title, and each transaction has a unique author ID followed by a set of such key-words (items). The data-mining provides information about the large sets of subjects(key-words) being researched. There are 13793 transactions, 10363 items, and the average transaction size is 22.

4.2 Accuracy Measurements

We report experimental results for the three databases described above. Figure 2 shows the number of large itemsets found during the different iterations of the *Apriori* algorithm. This is done for each database with different values of minimum support and sample size. In the graphs, ORIG indicates the actual number of large itemsets generated when the algorithm operates on the entire database. SAMP0.x refers to the large itemsets generated when using a sample of size x% of the entire database. ACT0.x refers to the number of itemsets generated by SAMP0.x that are large itemsets in the original database. The number of *false* large itemsets is given as (SAMP0.x − ACT0.x). We present results for x = 1%, 5%, 10%, and 20% (for TRBIB). Different minimum support values were chosen for the databases so that there were enough large k-itemsets (with $k > 5$). We used 0.25% and 0.75% for SYNTH, 0.75% and 1.0% for ENROLL, and 1.5% and 2.0% for TRBIB.

Salient Points

From figure 2 we can observe that the general trends of sampled databases resemble actual results for all values of minimum support. Smaller sample sizes tend to overestimate the number of large itemsets, i.e., they find more false large itemsets (with the exception of SYNTH at minimum support 0.75%). On the other hand, larger sample sizes tend to give better results in terms of fidelity (closeness of the curve to ORIG). This can be seen by the way ACT0.x comes closer to ORIG as x (the sample percentage) is increased. It is also interesting to note that with increasing minimum support the fidelity of sampling increases. As compared to ENROLL, for SYNTH we see that a sample size of 1% does relatively well. For ENROLL 1% sample size is inadequate. However, a 5% sample size gives quite good results. For TRBIB, even 5% is not sufficient (10% works well). It is clear that for smaller databases we need higher sampling sizes to generate large itemsets with reasonable fidelity.

The conclusion we can draw from the above points is that smaller values of sampling generate more large itemsets than ORIG, and a larger number of false large itemsets. With increasing sample sizes we come close to ORIG, and generate fewer false large itemsets. The graphs also suggest that beyond a certain sampling size there are only small gains in fidelity. The higher fidelity at higher sample size should

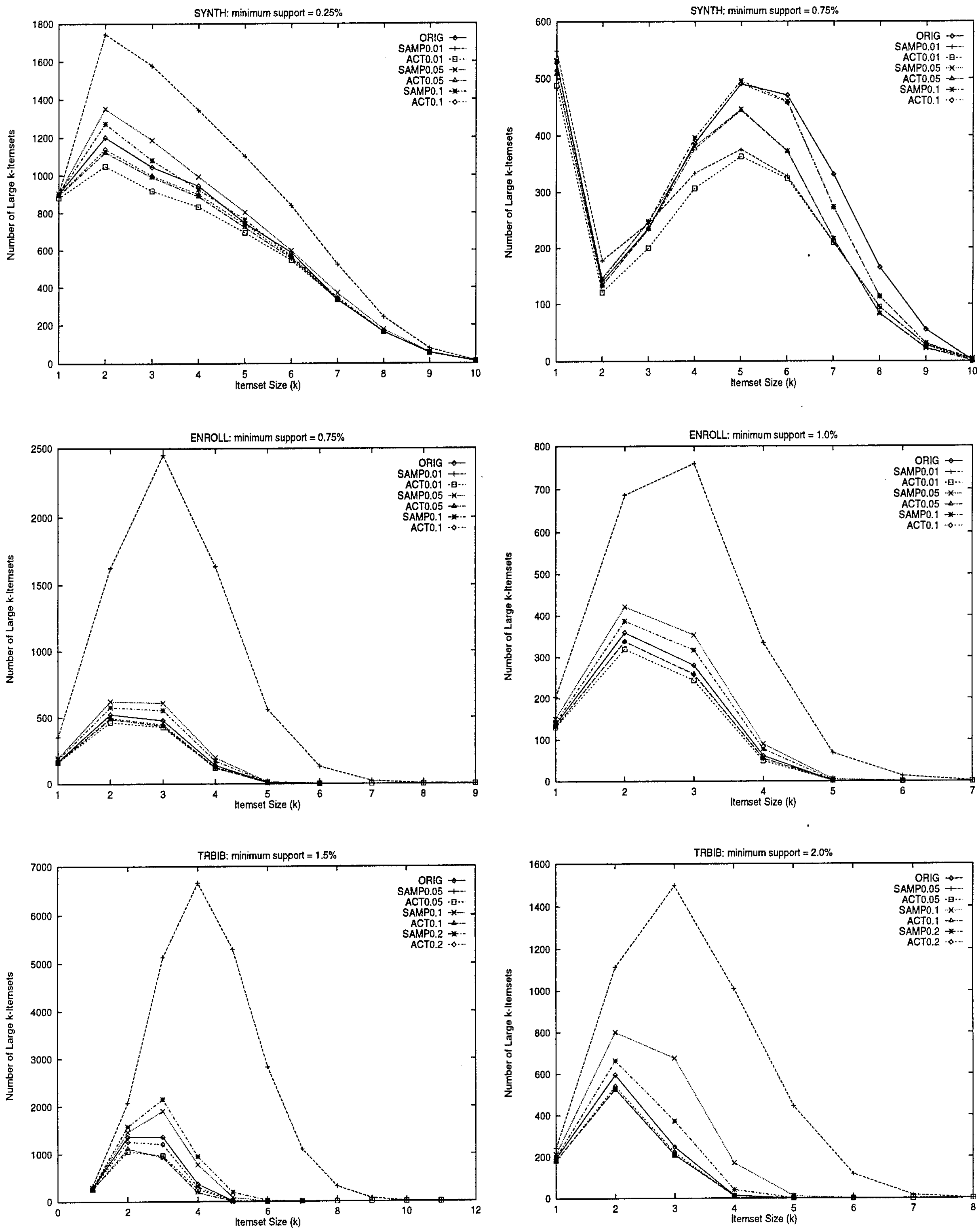


Figure 2: Itemset Size vs. Number of Large Itemsets

be weighed against the reduced performance gains (discussed below), while choosing a sample size.

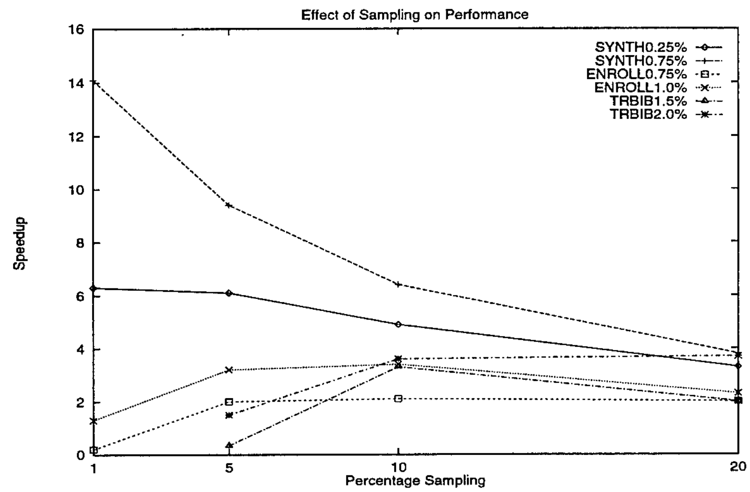


Figure 3: Sampling Performance

4.3 Performance

Figure 3 shows the speedup obtained for the databases on different minimum support and different sampling size values. The speedup is relative to the base algorithm execution time on the entire database. For SYNTH we obtain a speedup of more than an order of magnitude at smaller sample sizes. The speedup decreases as the sample size increases. For the smaller databases (ENROLL and TRBIB), at small sample size, we get no speedup, due to the large number of false large itemsets generated. We obtained a speedup between 2 and 4 at larger sample sizes. We can conclude that sampling is a very effective technique in terms of performance, and we can expect it to work very well with large databases, as they have higher computation and I/O overhead.

4.4 Confidence: Comparison with Chernoff Bounds

In this section we compare the Chernoff bound with experimentally observed results. We show that for the databases we have considered the Chernoff bound is very conservative.

Consider equations 1 and 2. For different values of accuracy, and for a given sampling size, for each itemset I, we can obtain the theoretical confidence value by simply evaluating the right hand side of the equations. For example, for the upper bound the confidence $\mathcal{C} = 1 - e^{-\epsilon^2 n\tau/3}$. Recall that confidence provides information about an item's actual support in the sample being away from the expected support by a certain amount ($n\tau\epsilon$). We can also obtain experimental confidence values as follows. We take s samples of size n, and for each item we compute the confidence by evaluating the left hand side of the two equations, as follows. Let $\imath$ denote the sample number, $1 \le \imath \le s$. Let

$$l_I(\imath) = \begin{cases} 1 & \text{if}(n\tau - \mathbf{X}) \ge n\tau\epsilon \text{ in sample } \imath \\ 0 & \text{otherwise} \end{cases}$$

$$h_I(\imath) = \begin{cases} 1 & \text{if}(\mathbf{X} - n\tau) \ge n\tau\epsilon \text{ in sample } \imath \\ 0 & \text{otherwise} \end{cases}$$

The confidence can then be calculated as $1 - \sum_{\imath=1}^{m} h_I(\imath)/s$, for the upper bound, and $1 - \sum_{\imath=1}^{m} l_I(\imath)/s$, for the lower bound.

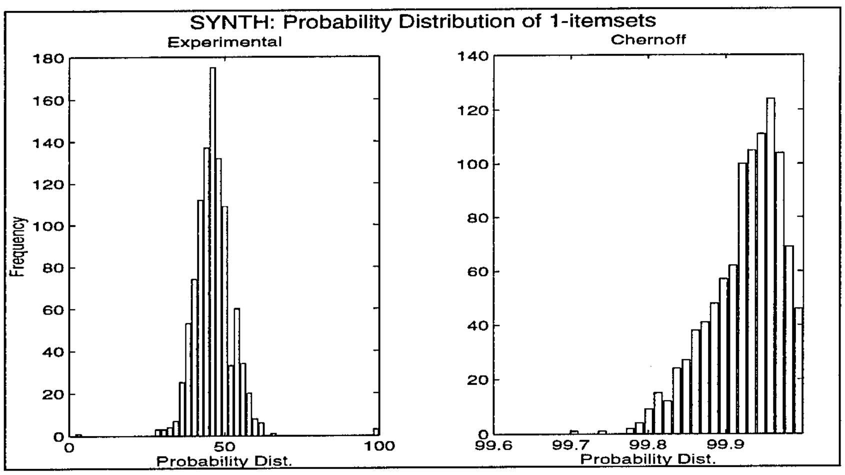


Figure 4: Probability Distribution: Experiment vs. Chernoff

Figure 4 compares the distribution of experimental confidence to the one obtained by Chernoff upper bounds, for all m 1-itemsets or single items. It is possible (though impractical) to do this analysis for all the 2^m itemsets, however we present results for only single items. This should give us an indication whether the sample faithfully represents the original database. The results shown are for the SYNTH database with $\epsilon = 0.01$, $n = 2500$ (1% of total database size), and the number of samples taken, $s = 100$. We can see that the probability distribution across all items varies from 0.30 to 0.60 for the experimental case, with a mean probability close to 0.43. The Chernoff bounds produce a distribution clustered between 0.998 and 1.0, with an average probability of 0.9992. Chernoff bounds indicate that it is very likely that the sample doesn't have the given accuracy, i.e., with high probability, the items will be overestimated by a factor of 1.01. However, in reality, the probability of being overestimated is only 0.43. The obvious difference in confidence depicts the limitation of

Chernoff bounds in this setting. This was observed in all three of the databases we looked at.

Figure 5 gives a broader picture of the large gap between Chernoff bounds and experimentally obtained effectiveness of sampling. For all three databases we plot the mean of the confidence or probability distribution for different accuracies $(1-\epsilon)$. The mean confidence obtained from Chernoff bounds is marked as T.x, and that obtained experimentally is marked as E.x. Different values of the sample size x are plotted (from 1% to 50%), and results for both the upper and lower bounds are shown. For all the three databases the upper and lower bounds give similar results. There is a small difference in the Chernoff bound values due to the asymmetry in equations 1 and 2. This is also true for the experimental results. For both cases the lower bounds give a slightly higher confidence for the same value of accuracy, as expected from the Chernoff bounds.

For SYNTH we observe that as the accuracy is compromised (as ϵ increases) the mean confidence across all items increases exponentially (therefore, only ϵ values upto 0.5 are shown). Furthermore, as the sample size increases, the curve falls more rapidly, so that we have higher confidence even at relatively higher accuracies. For both ENROLL and TRBIB we get the same general trends, however the increase in confidence for lower accuracies is not as rapid. This is precisely what we expect. For example, consider the right hand side of Chernoff lower bounds (equation 1), $e^{-\epsilon^2 n\tau/2} = \mathcal{L}$. For a given ϵ and τ (the support for an item), a higher value of n gives us high confidence, as it results in a lower value for $\mathcal{L}$. For a given sampling percentage, since SYNTH is large, we expect a higher confidence than that for ENROLL or TRBIB (for example, with sampling = 10%, $\epsilon = 0.1$, and $\tau = 0.01$, we get $n = 25000$, $\mathcal{L} = 0.29$ for SYNTH, $n = 3962$, $\mathcal{L} = 0.82$ for ENROLL, and $n = 1379$, $\mathcal{L} = 0.93$ for TRBIB). We get the same effect for the experimental results.

We can observe that for all three databases, the experimental results predict a much higher confidence, than that using Chernoff bounds. Furthermore, from the above analysis we would expect sampling to work well for larger databases. The distribution of the support of the itemsets in the original database also influences the sampling quality.

5 Conclusions

We have presented experimental evaluation of sampling for three separate databases to show that it can be an effective tool for data mining. The experimental results indicate that sampling can result in not only performance savings (such as reduced I/O cost and total computation), but also good accuracy (with high confidence) in practice, in contrast to the confidence obtained by applying Chernoff bounds. Therefore, we claim that for practical purposes we can use sampling with confidence for data mining.

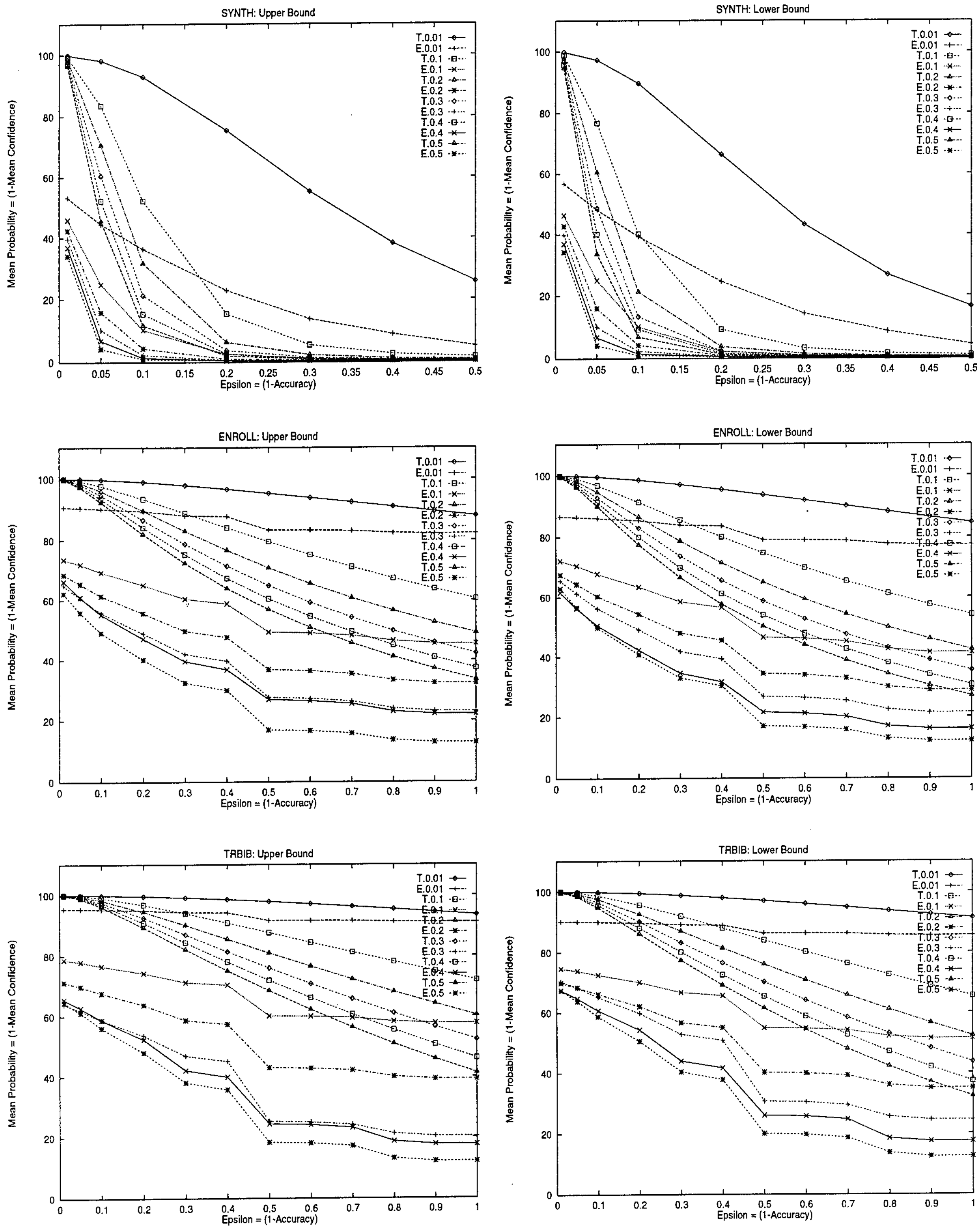


Figure 5: Accuracy vs. Mean Confidence for Single Items

References

[1] R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In *Proc. ACM SIGMOD Intl. Conf. Management of Data*, May 1993.

[2] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proc. 20th VLDB Conf.*, Sept. 1994.

[3] W. G. Cochran. *Sampling Techniques.* John Wiley & Sons, 1977.

[4] T. Hagerup and C. Rüb. A guided tour of chernoff bounds. In *Information Processing Letters*, pages 305–308. North-Holland, 1989/90.

[5] M. Holsheimer, M. Kersten, H. Mannila, and H. Toivonen. A perspective on databases and data mining. In *1st Intl. Conf. Knowledge Discovery and Data Mining*, Aug. 1995.

[6] M. Houtsma and A. Swami. Set-oriented mining of association rules. In *RJ 9567.* IBM Almaden, Oct. 1993.

[7] H. Mannila, H. Toivonen, and I. Verkamo. Efficient algorithms for discovering association rules. In *AAAI Wkshp. Knowledge Discovery in Databases*, July 1994.

[8] F. Olken and D. Rotem. Random sampling from databases - a survey. In *Draft.* ICS Div, Lawrence Berkeley Lab., Mar. 1994.

[9] J. S. Park, M. Chen, and P. S. Yu. An effective hash based algorithm for mining association rules. In *Proc. ACM SIGMOD Intl. Conf. Management of Data*, May 1995.

[10] A. Savasere, E. Omiecinski, and S. Navathe. An efficient algorithm for mining association rules in large databases. In *Proc. 21st VLDB Conf.*, 1995.

[11] J. S. Vitter. An efficient algorithm for sequential random sampling. Technical Report 624, INRIA, Feb. 1987.